This listing of claims will replace all prior versions, and listings, of claims in the application.

## Listing of Claims:

1-14. (Canceled)

15. (Original) A method for scheduling tasks for processing by a coprocessor, comprising:

gathering tasks for processing by a coprocessor into a memory group wherein the memory group relates to a first application;

delivering the tasks to a scheduler wherein scheduler functions include determining an order for processing the tasks wherein the order may include tasks that relate to one or more other applications;

determining an order for processing the tasks wherein the order accounts for any relative priority among the first application and one or more other applications and a corresponding amount of processing time that the first application and one or more other applications are entitled to;

preparing tasks for processing by ensuring that any needed memory resources are available in a coprocessor-accessible memory location wherein the preparing tasks occurs in the order determined by the scheduler; and

submitting tasks prepared according to the preparing to the coprocessor for processing.

16. (Original) A method according to claim 15 wherein the coprocessor includes a graphics processing unit (GPU).

17. (Original) A method according to claim 15, further comprising calling an Application Program Interface (API) when the first application has one or more tasks that require processing by the coprocessor.

18. (Original) A method according to claim 17, further comprising calling a user mode driver wherein the functions of the user mode driver include placing rendering commands associated with the one or more tasks in the memory group.

19. (Original) A method according to claim 18, further comprising returning the rendering commands to the API, and submitting them to a coprocessor kernel.

20. (Previously Presented) A method according to claim 15, further comprising generating a Direct Memory Access (DMA) buffer by a kernel mode driver wherein one or more tasks that require processing by the coprocessor are used to generate the DMA buffer, and the DMA buffer represents the one or more tasks used to generate the DMA buffer.

21. (Original) A method according to claim 20, further comprising generating a list of memory resources by the kernel mode driver wherein the memory resources represented by the list are needed by the coprocessor to process one or more tasks represented by the DMA buffer.

22. (Original) A method according to claim 21, further comprising building a paging buffer for bringing the memory resources on the list of memory resources to correct memory addresses within the coprocessor-accessible memory location.

23. (Original) A method according to claim 15 wherein said preparing is accomplished by a preparation thread which calls a memory manager process capable of determining a location in the coprocessor-accessible memory location to page any needed memory resources.

24. (Original) A method according to claim 23, further comprising splitting a DMA buffer when the memory manager process determines that there is not enough room in the coprocessor-accessible memory location to page all needed memory resources.

25. (Original) A computer readable medium comprising computer executable instructions for carrying out the method of claim 15.

26. (Original) A modulated data signal carrying computer executable instructions for use in performing the method of claim 15.

27. (Original) A computing device comprising means for performing the method of claim 15.

28. (Original) A method for scheduling tasks for processing by a coprocessor, comprising:

gathering tasks for processing by a coprocessor into a memory group wherein the memory group relates to a first application;

delivering the tasks to a scheduler wherein the functions of the scheduler include determining an order for processing the tasks wherein the order may include tasks that relate to one or more other applications;

determining an order for processing the tasks wherein the order accounts for any relative priority among the first application and one or more other applications and a corresponding amount of processing time that the first application and one or more other applications are entitled to;

preparing tasks for processing by ensuring that any needed memory resources are available in a memory location accessible by the coprocessor wherein the preparing tasks occurs in the order determined by the scheduler;

submitting tasks to the coprocessor for processing;

managing the coprocessor-readable memory to apportion the coprocessor-readable memory among the various tasks; and

providing a virtual address space for the tasks.

29. (Original) A method according to claim 28 wherein the coprocessor is a graphics processing unit (GPU).

30. (Original) A method according to claim 28, further comprising storing a task in a DMA buffer wherein the storing is accomplished by a user mode driver.

31. (Original) A method according to claim 30, further comprising validating a memory resource referenced in a resource list that is associated with the DMA buffer wherein validating entails finding a range of coprocessor-readable memory that is free and asking the kernel mode driver to map a page table or a memory resource handle to that range.

32. (Original) A method according to claim 28 wherein the virtual address space is virtualized through the use of a flat page table that divides coprocessor-readable memory into

pages of a predefined memory amount wherein further a page table is provided in the virtual address space that contains identifiers for specifying coprocessor-readable memory addresses.

33. (Original) A method according to claim 28 wherein the virtual address space is virtualized through the use of a multi-level page table that divides coprocessor-readable memory into pages of a predefined memory amount wherein further a multiple page tables are provided in the virtual address space that contain identifiers for specifying coprocessor-readable memory addresses.

34. (Original) A method according to claim 28 wherein a portion of coprocessor readable memory is used to indicate whether all required memory resources associated with a task that requires processing are available in coprocessor-readable memory.

35. (Original) A computer readable medium comprising computer executable instructions for carrying out the method of claim 28.

36. (Original) A modulated data signal carrying computer executable instructions for use in performing the method of claim 28.

37. (Original) A computing device comprising means for performing the method of claim 28.

38. (Original) A method according to claim 28, further comprising:

assigning a base address for a display surface wherein the display surface is allocated contiguously in coprocessor-readable memory; and

delivering a task to the scheduler wherein processing the task will reassign the base address for a display surface.

39. (Original) A method according to claim 38 wherein processing the task will reassign the base address for a display surface immediately.

40. (Original) A method according to claim 38 wherein processing the task will reassign the base address for a display surface upon the occurrence of a subsequent display synchronization period.

41. (Original) An apparatus for supporting scheduling of tasks for processing by a coprocessor, comprising:

a central processing unit (CPU);

a coprocessor;

one or more applications that generate tasks for processing by the coprocessor wherein the tasks are first stored in an application-specific memory location;

a scheduler process for determining an order in which the tasks are processed; wherein the order accounts for any relative priority among a first application and one or more other applications and a corresponding amount of processing time that the first application and one or more other applications are entitled to.

42. (Original) An apparatus according to claim 41 wherein the coprocessor is a GPU.

43. (Original) An apparatus according to claim 41 wherein the coprocessor supports interruption during the processing of a task by automatically saving task information to a coprocessor-accessible memory location.

44. (Original) An apparatus according to claim 43, further comprising at least one of a private address space for one or more tasks, a private ring buffer where tasks are accumulated, and a private piece of coprocessor-accessible memory where a hardware state is saved when a task is not being processed.

45. (Original) An apparatus according to claim 41 wherein the coprocessor is capable of storing information regarding the history of coprocessor switches from task to task in a specified system memory location readable by the scheduler process.

46. (Original) An apparatus according to claim 45 wherein the coprocessor specifies a base address for the system memory location prior to storing information regarding the history of coprocessor switches from task to task in the system memory location.

47. (Original) An apparatus according to claim 45 wherein the coprocessor specifies a size for the system memory location prior to storing information regarding the history of coprocessor switches from task to task in the system memory location.

48. (Original) An apparatus according to claim 45 wherein the coprocessor specifies a write pointer for indicating where in the system memory location the coprocessor should write to next.

49. (Original) An apparatus according to claim 41 wherein the coprocessor supports fence instructions that cause the coprocessor to write a piece of data associated with a fence instruction at an address specified in the fence instruction.

50. (Original) An apparatus according to claim 41 wherein the coprocessor supports trap instructions that are capable of generating a CPU interrupt when processed by the coprocessor.

51. (Original) An apparatus according to claim 41 wherein the coprocessor supports enable/disable context switching instructions such that when context switching is disabled, the coprocessor will not switch away from a current coprocessor task.

52-65. (Canceled)

66. (Original) A coprocessor for use in connection with a coprocessing scheduler, comprising:

a coprocessor for processing tasks that are submitted to the coprocessor by a scheduler process wherein the scheduler process submits tasks to the coprocessor according to a priority of applications that request processing of the tasks, and wherein the priority determines the amount of coprocessor time one or more applications are entitled to.

67. (Original) A coprocessor according to claim 66 wherein the tasks are first stored in an application-specific memory location.

68. (Original) A coprocessor according to claim 66 wherein the coprocessor stores information related to a task in a per-context address space, and wherein further the

information related to a task allows the coprocessor to process the task or a portion of the task after processing one or more intervening tasks.

69. (Original) A coprocessor according to claim 66 wherein the coprocessor processes tasks from a run list by switching immediately to a subsequent task on the run list when a switching event occurs.

70. (Original) A coprocessor according to claim 69 wherein a switching event comprises at least one of a completion of processing a previously submitted task, a page fault in processing a task, a general protection fault in processing a task, and a request by a central processing unit (CPU) to switch to a new run list.

71. (Original) A coprocessor according to claim 66 wherein the coprocessor comprises a GPU.

72. (Original) A coprocessor according to claim 66 wherein the coprocessor accesses memory resources in a coprocessor-readable memory by a memory manager.

73. (Original) A coprocessor according to claim 72 wherein the memory resources comprise references to virtual memory addresses.

74-79. (Canceled)